Algoritmos Evolutivos - Estudio de Escalabilidad sobre el Problema COUNTSAT

Hugo Alfredo Alfonso
Facultad de Ingeniería
Universidad Nacional de La Pampa
La Pampa, Argentina
email: alfonsoh@ing.unlpam.edu.ar

Enrique Alba
Escuela Técnica Superior de Ingeniería Informática
Universidad de Málaga
Málaga, España
email: eat@lcc.uma.es

Resumen

En este trabajo se investiga la influencia del tamaño del problema en la performance de los Algoritmos Evolutivos (AEs), utilizados para resolver una variante NP-difícil del problema MAXSAT denominada COUNTSAT. Para ello se ha realizado una recopilación de diferentes tipos de AEs, desarrollados para mejorar la calidad de los resultados. También, se describen los AEs usados para analizar el comportamiento de los mismos frente a problemas de tamaño creciente, tomados de benchmarks internacionales. Este estudio incluye el análisis de la incorporación de distribución y paralelismo en la resolución del problema, como caminos alternativos para encontrar el óptimo con menor esfuerzo numérico.

Palabras Claves: Algoritmo Evolutivo Distribuido, Problema SAT
Evento: Workshop de Agentes y Sistemas Inteligentes

1. Introducción

El *Problema de Satisfacción de cláusulas lógicas (SAT)* es un paradigmático problema NP-completo [Coo71], que también cuenta con un muy importante interés práctico, pues se aplica en problemas de razonamiento, diagnóstico y planificación [KS92], interpretación de imágenes [RM89], entre otros. El Problema SAT está basado sobre un conjunto de variables lógicas $x_1, x_2, \ldots, x_n$ y una fórmula lógica $f : \mathcal{B}^n \rightarrow \mathcal{B}, \mathcal{B} = \{0, 1\}$. Siendo la cuestión a resolver la existencia de una asignación de valores posibles $x = (x_1, x_2, \ldots, x_n) \in \mathcal{B}^n$ que satisfagan dichas fórmula, es decir que $f(x) = 1$. Una instancia SAT se denomina *satisfecha* si existe tal valor x, e *insatisfecha* en caso contrario. La fórmula f está en *Forma Normal Conjuntiva - (Conjunctive Normal Norm-CNF)* si $f(x) = c_1(x) \wedge \ldots \wedge c_m(x)$, donde cada una de las cláusulas c_i es una disjunción de literales, y un literal es una variable o su negación. Se puede asumir que las instancias SAT están expresadas en CNF sin pérdida de generalidad [S.68], y se suelen definir las clases l-SAT como instancias del problema cuyas cláusulas contienen exactamente l literales distintos. La clase de problemas 2-SAT es resoluble en tiempo polinomial, pero las clases l-SAT son NP-completas para $l \geq 3$ [GJ79].

Debido a la importancia teórica y práctica del problema SAT, y particularmente el 3-SAT, este ha sido ampliamente estudiado y se han desarrollado para él muchos algoritmos exactos y heurísticos. De acuerdo a la compilación que realizase Jun Gu et al. [GPFW96] varios de los algoritmos dan una respuesta definitiva a la pregunta inicialmente planteada (si cada instancia del problema es satisfecha o no), pero tiene una complejidad exponencial al peor caso, a menos que $P = NP$. Por el contrario los algoritmos heurísticos pueden encontrar soluciones para instancias satisfechas rápidamente, pero no pueden garantizar el dar una respuesta para todas las instancias del problema.

Los *Algoritmos Evolutivos (AEs)* son algoritmos heurísticos que se han aplicado al problema SAT, como así también a otros tantos problemas NP-completos. Algunos resultados negativos ponen en duda la capacidad de los AEs para resolver el problema SAT. De Jong y Spears [DS89] propusieron un *Algoritmo Genético (AG)* clásico para el SAT y observaron que un AG no podía obtener mejores resultados que los algoritmos específicos perfectamente adecuados al problema. Este resultado fue confirmado por las experiencias de Fleurent y Ferland [FF93], quienes informaron un escaso rendimiento del AG puro cuando es comparado con búsqueda local. Posteriormente, Rana y Whitley [RW98] demostraron que los AGs clásicos no eran lo suficientemente apropiados para la función de fitness del problema de maximización *MAXSAT*, la cual cuenta el número de cláusulas satisfechas, por que el dominio correspondiente al problema presenta engañosos esquemas de información de baja importancia y el espacio de búsqueda tiende a ser homogéneo. Sin embargo, estudios más recientes muestran que los AEs pueden obtener buenos resultados para el problema SAT si se le incorporan técnicas adicionales con el objetivo de mejorar los AGs clásicos. Estas técnicas incluyen el uso de funciones de fitness adaptativas, operadores de variación específicos al problema, optimización local, paralelización, etc. [DS89] [BEV98] [EvdH97] [FPS98] [MR99].

El paralelismo y la distribución ofrecen ventajas cuando los AEs trabajan con pobla-

ciones de gran tamaño y/o soluciones tentativas de considerable longitud, lo cual supone una significativa utilización de recursos computacionales, como memoria física y tiempo de procesador. El esfuerzo computacional se puede disminuir al ejecutar el algoritmo en un ambiente distribuido. Los modelos de estos algoritmos que permiten la distribución del procesamiento ya sea sobre uno o varios procesadores se denominan Algoritmos Evolutivos Distribuidos (AEd) o Algoritmos Evolutivos Paralelos (AEp) (ver pseudocódigo en Cuadro 1).

```
AEd^i
  t^i := 0
  inicializar [P^i(t^i)]
  evaluar [P^i(t^i)]
  MIENTRAS no cumpla Criterio Terminacion HACER
      P'^i(t^i) := evolucion[P^i(t^i)] -- crossover y mutacion
      evaluar[P'^i(t^i)]
      P'^i(t^i) := comunicacion[P^i'(t^i) ⋃ AEd^j :: P^j(t^j)] -- intercambio con proxima isla
      P^i(t^i + 1) := seleccionar nuevo entorno[P^i(t^i) ⋃ P'^i(t^i)]
          -- seleccionar individuos para proxima generacion
      t^i := t^i + 1
  FIN MIENTRAS
```

Cuadro 1: Pseudocódigo de AE distribuido para el proceso i

Ellos permiten modificar el comportamiento típico del algoritmo secuencial mediante el uso de una población estructurada (una distribución espacial de individuos ya sea en la forma de un conjunto de islas [Tan89] o de una rejilla de difusión [MS89],[PB91]).

En este trabajo se analiza el comportamiento de los AEs al resolver el problema COUNTSAT frente a instancias cada vez más complejas lo que permite evaluar su performance frente al crecimiento del tamaño del problema. Por otra parte se analizará el comportamiento en distintos ambientes, uno distribuido secuencial y otro distribuido paralelo, para poder identificar similitudes y/o diferencias en los algoritmos resultantes.

Este trabajo está organizado de la siguiente forma: primero se define el problema COUNTSAT usado para hacer este estudio, luego se introducen algunos tópicos relacionados con los modelos de distribución considerados. En la Sección 4 se describen las instancias y variables que miden la performance alcanzada durante la experimentación; experimentación que se detalla en la Sección 5. En la Sección 6 se discuten los resultados obtenidos. En la Sección 7 se estudia si existe o no influencia del tamaño de la población en la performance de los algoritmos para finalmente extraer conclusiones en la Sección 8.

2. Problema COUNTSAT

En este trabajo se ha realizado la experimentación sobre una variante del problema MAX-SAT, denominada COUNTSAT [DJW00], la cual ha sido presentada en [Pap94]. Para ello se asume que:

- $x_i, 1 \leq i \leq n$ (hay n variables)

- $x_i \vee \overline{x}_j \vee \overline{x}_k, (i,j,k) \in \{1,\ldots,n\}^3, i \neq j \neq k \neq i$ (es una variante 3-SAT)

En COUNTSAT, el valor de la solución es el número de cláusulas que son satisfechas por una determinada cadena de entrada de n bits. Es relativamente fácil chequear que el valor óptimo se obtiene con la cadena de entrada donde todos los n bits tengan el valor 1.

La función objetivo del problema COUNTSAT puede ser definida de la siguiente forma:

$$f_{COUNTSAT}(s) = s + n \cdot (n-1) \cdot (n-2) - 2 \cdot (n-2) \cdot \binom{s}{2} + 6 \cdot \binom{s}{3} \quad (1)$$

Esta función, al igual que todas aquellas que surgen del problema SAT, tiene la particularidad de ser muy dura a medida que la cantidad de variables se incrementa.

3. Paralelismo y Distribución

Las características interesantes que incluyen los AEp o AEd son: (a) disminución del tiempo para localizar una solución (algoritmos más rápidos), (b) reducción en el número de evaluaciones (menor costo de búsqueda), (c) posibilidad de tener poblaciones de mayor tamaño y (d) aumento de la calidad en las soluciones halladas. Las versiones de AEp y AEd son menos propensas a la convergencia prematura con lo cual se mejora el proceso de búsqueda [AT99], no son simplemente "versiones más rápidas"de AEs secuenciales, sino que además proporcionan un mecanismo de búsqueda distinto, frecuentemente mejor. Ellos extienden la versión secuencial incluyendo una fase de comunicación con un vecindario formado por otros AEs. Una forma de lograr la paralelización es considerar un modelo de selección descentralizado donde los individuos se agrupan espacialmente. Este agrupamiento resulta de particionar una única población en varias y da origen a islas de AEs que se ejecutan realizando intercambios esporádicos de individuos. Dichas islas pueden utilizar: (a) algoritmos cuya evolución básica es secuencial y panmíctica (por ejemplo [WS90]), y (b) algoritmos celulares o de otro tipo. El tipo de AE distribuido en uso lo determina la política de migración, la cual define: la topología de las islas, los individuos a intercambiar, la sincronización entre las subpoblaciones y la forma en que se integran los individuos en la subpoblación destino. Existe un consenso respecto a utilizar una topología anillo (o hipercubo) debido a su simplicidad, fácil implementación y ganancia en tiempo de comunicación.

4. Instancias de Testeo y Variables de Performance

En la experimentación desarrollada para este trabajo se ha considerado la función COUNTSAT variando la cantidad n de variables entre 1 y 30. Se muestra en la Figura 1 cómo crece el valor óptimo dependiendo del número de variables (por tanto el número de soluciones posibles al problema SAT, que es lo que representa el COUNTSAT, aunque existe una sola satisfacible).

Las variables que permiten medir el comportamiento de los algoritmos son: SR (Success Rate) y AES (Average number of Evaluations to Solution) las cuales han sido reportadas en diversos trabajos del área como [JK] [MR99] [GV98] [GMR02]:

SR indica el porcentaje del total de experimentos en los cuales la solución óptima ha sido alcanzada (tasa de éxitos)

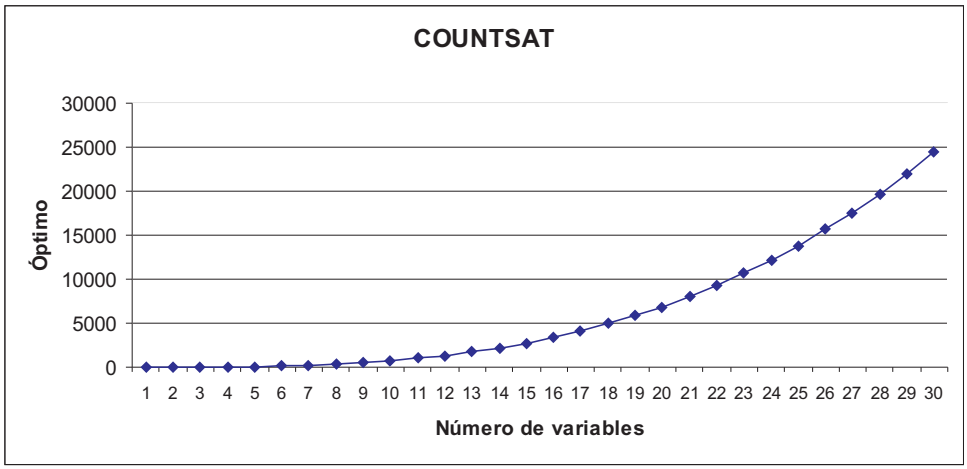

Figura 1: Identificación de valores óptimos de la función COUNTSAT.

AES indica el número de evaluaciones necesarias para terminar aquellos experimentos que alcanzan la solución óptima (esfuerzo o costo de calcular una solución)

Es claro que, si $SR = 0$ entonces AES no está definido.

5. Detalle de Experimentos

El Algoritmo Evolutivo utilizado para realizar las pruebas pertenece al paquete MALLBA que fue desarrollado como parte del Proyecto Mallba [AAB+01] por investigadores de tres Universidades Españolas: **MÁ**laga, **L**a **L**aguna y **BA**rcelona.

En el proyecto MALLBA se ataca la resolución de diversos problemas de optimización combinatoria usando esqueletos algorítmicos desarrollados en C++ sobre una red de PCs conectadas en una red geográficamente distribuida. Además, MALLBA ofrece tres familias de métodos de resolución genérica de problemas: exactos, heurísticos e híbridos. Para cada uno de estos métodos ofrece tres variantes de implementación: secuencial, LAN y WAN.

Para la experimentación se han realizado series de 50 ejecuciones independientes sobre cada una de las instancias mencionadas en la sección anterior. Cada instancia ha sido tratada con tres algoritmos, los cuales son:

- AE - Algoritmo Evolutivo Panmíctico
- AEd - Algoritmo Evolutivo Distribuido sobre un procesador
- AEp - Algoritmo Evolutivo Distribuido sobre 4 procesadores

Los AEs usan la selección por torneo binario para hacer la selección del conjunto de padres, durante el proceso evolutivo se aplica el crossover de un punto (One Point Crossover) con una probabilidad $Pc = 0{,}65$ y a cada hijo generado se le aplica Big-Creep Mutation con probabilidad $Pm = 0{,}01$. Los valores de los parámetros fueron seleccionados como la mejor combinación de probabilidades luego de varias pruebas iniciales.

El tamaño de la población siempre se ha mantenido constante durante el proceso evolutivo. El caso base utiliza una población de 128 individuos, y genera una población de hijos de igual tamaño que se combina con la población de padres para seleccionar los 128 mejores individuos que pasan a la próxima generación.

Para el AE Distribuido (AEd) se usó un modelo de cuatro procesos asíncronos trabajando sobre el mismo computador. El modelo isla realiza una migración cada 25 generaciones, enviando 5 individuos cada vez considerando una topología de anillo. El grupo de individuos migrados que cada isla recibe es inspeccionado para insertar en la población actual aquellos individuos recibidos que sean mejores que el peor individuo actual de la isla.

Para el AE Paralelo (AEp), se usa el mismo modelo descripto para AEd pero trabajando en 4 computadores diferentes.

En ambos casos el tamaño de la población de cada isla se fijó en un cuarto de la población total usada en panmixia, para mantener el tamaño total de la población siempre fija.

Los detalles de los parámetros se describen en el Cuadro 2.

	AE	**AEd**	**AEp**
Número de Procesadores	1	1	4
Número de Procesos	1	4	4
Tamaño de población	128	32	32
Crossover	Crossover de un punto con $P_c = 0{,}65$		
Mutación	Big-Creep Mutation con probabilidad $Pm = 0{,}01$		
Criterio de parada	Alcanzar óptimo o 100.000 generaciones		
Topología de red	-	Anillo unidireccional estático	
Frecuencia de Migración	-	Cada 25 generaciones	
Número de Individuos que Migran	-	5	
Política de Selección de Emigrantes	-	Selecciona los mejores	
Política de Reemplazo de Inmigrantes	-	Reemplaza a los peores sólo si son mejores	

Cuadro 2: Parámetros de algoritmos utilizados.

6. Resultados

Las figuras y cuadros que a continuación se presentan resumen los resultados alcanzados por los algoritmos descriptos.

Analizando la Figura 2 que muestra el porcentaje de éxito obtenido por cada una de las variantes algorítmicas sobre las instancias consideradas, es claro que los algoritmos alcanzan un 100 % de éxito para las primeras 20 instancias (SR = 1). Las siguientes instancias gradualmente van provocando en los tres algoritmos mayor dificultad en alcanzar el valor óptimo, agudizándose tal situación desde la instancia 25, ello puede ser corroborado en los resultados que se muestran en el Cuadro 3. Parece existir una transición de fase en la dificultad en el rango 20-30 variables.

Si analizamos el número de evaluaciones necesarias para alcanzar la solución óptima (AES), en el Cuadro 3, podemos observar el incremento según aumentemos la complejidad al problema (mayor n). Al compararlo sobre la Figura 3 puede observarse que el algoritmo que se ejecuta en forma panmíctica (AE) requiere mayor cantidad de evaluaciones que aquellos que distribuyen el trabajo, ya sea en uno (AEd) o cuatro (AEp) procesadores. En promedio estos últimos requieren sólo entre el 20 % y 30 % del número de evaluaciones que necesita el AE, siendo superior el desempeño del AEd. En la Figura 3 se han señalado las instancias que son consideradas más duras

Inst.	AE		AEp		AEd	
	SR	AES	SR	AES	SR	AES
1	1.00	256.0	1.00	64.0	1.00	64.0
2	1.00	256.0	1.00	64.0	1.00	64.0
3	1.00	256.0	1.00	67.8	1.00	65.9
4	1.00	256.0	1.00	75.5	1.00	73.6
5	1.00	263.7	1.00	142.7	1.00	110.1
6	1.00	340.5	1.00	188.8	1.00	263.7
7	1.00	578.6	1.00	277.1	1.00	559.4
8	1.00	1,484.8	1.00	542.1	1.00	1,465.6
9	1.00	3,850.2	1.00	1,037.4	1.00	2,796.2
10	1.00	6,384.6	1.00	1,636.5	1.00	4,735.4
11	1.00	13,619.2	1.00	3,429.8	1.00	7,365.8
12	1.00	26,775.0	1.00	5,113.6	1.00	11,662.7
13	1.00	49,753.6	1.00	17,378.6	1.00	18,931.8
14	1.00	140,938.2	1.00	33,089.9	1.00	25,845.8
15	1.00	260,869.1	1.00	58,387.8	1.00	54,083.2
16	1.00	459,189.8	1.00	94,650.9	1.00	118,783.4
17	1.00	657,617.9	1.00	158,368.0	1.00	245,893.1
18	1.00	1,615,429.1	1.00	573,454.7	1.00	444,574.7
19	1.00	4,278,599.7	1.00	1,019,345.9	1.00	1,047,721.6
20	1.00	7,403,745.3	0.98	1,810,577.0	0.98	1,677,654.2
21	0.88	12,144,683.6	0.94	3,540,811.6	0.92	3,008,729.0
22	0.74	11,697,248.9	0.78	3,475,342.8	0.76	2,630,762.1
23	0.52	16,836,839.4	0.52	4,502,216.6	0.46	4,401,977.0
24	0.30	18,573,209.6	0.24	6,515,168.0	0.44	4,590,430.5
25	0.18	18,642,176.0	0.12	4,248,960.0	0.14	4,725,773.7
26	0.10	15,124,940.8	0.04	2,484,160.0	0.14	3,999,273.1
27	0.04	17,575,744.0	0.04	5,431,360.0	0.02	5,059,360.0
28	0.00	0.0	0.02	4,015,360.0	0.00	0.0
29	0.00	0.0	0.00	0.0	0.02	2,522,560.0
30	0.02	1,793,536.0	0.00	0.0	0.00	0.0

Cuadro 3: Performance de los algoritmos usados sobre todas las instancias consideradas.

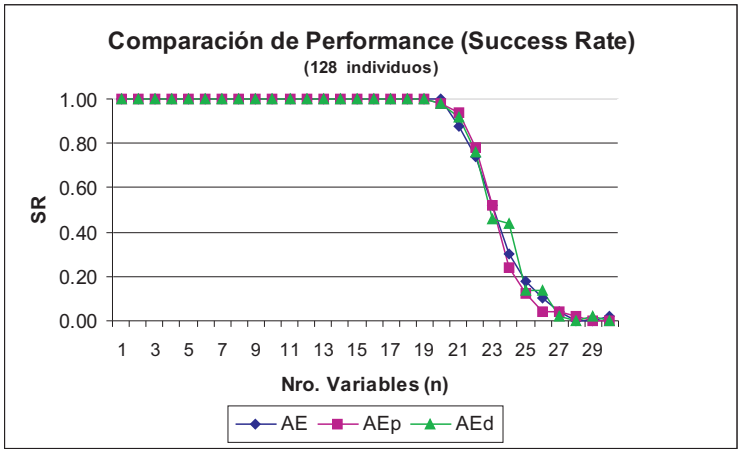

Figura 2: Análisis de SR sobre cada una de las instancias.

y que en muchos de los casos no alcanzan el valor óptimo, lo que en cierta medida distorsiona el crecimiento del AES por ser cero el valor correspondiente cuando $SR = 0$.

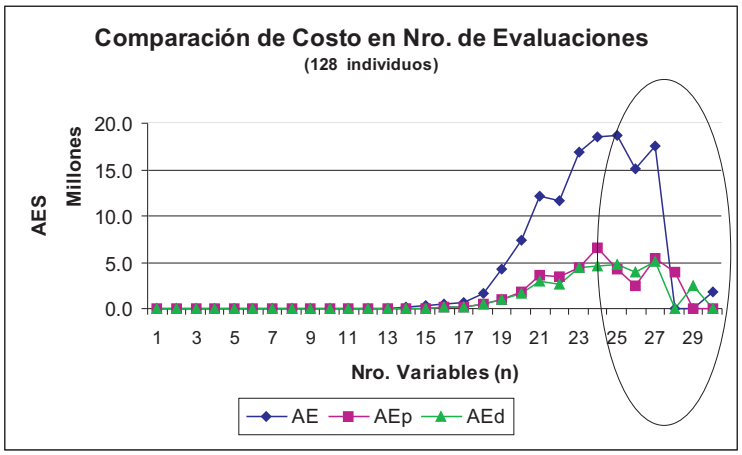

Figura 3: Análisis de AES sobre cada una de las instancias.

7. Influencia del Tamaño de la Población en los Resultados

Dado que uno de los objetivos planteados en el trabajo es analizar la escalabilidad de los algoritmos, independientemente del tamaño de la población, se ha replicado la experimentación para evaluar dos tamaños de población más, uno que considere la mitad de la población original (64 individuos) y otro que la duplique (256 individuos).

En el Cuadro 4 se presenta el promedio obtenido sobre la totalidad de las instancias evaluadas, considerando las tres variantes algorítmicas que se han contemplado. Es muy fácil observar que la calidad de los resultados (SR) mejora al incrementar el tamaño de la población, pero esa mejora es poco significativa frente al esfuerzo medido en cantidad de evaluaciones necesarias (AES).

Tamaño de Población	**AE**		**AEp**		**AEd**	
	SR	**AES**	**SR**	**AES**	**SR**	**AES**
64	0.73	2,846,491.31	0.72	757,334.04	0.73	599,490.36
128	0.76	4,243,628.05	0.76	1,266,375.70	0.76	1,153,385.99
256	0.79	9,880,481.99	0.80	2,588,421.99	0.79	2,529,192.99

Cuadro 4: Performance promedio de los algoritmos usados sobre la totalidad de las instancias evaluadas considerando diversos tamaños de población.

Las Figuras 4 y 5 nos muestran las variaciones de performance para los dos nuevos tamaños de población considerados, que junto a las gráficas previamente presentadas se puede observar un similar comportamiento, de ello se desprende que el comportamiento de los algoritmos se mantiene independientemente del tamaño de la población. Por lo tanto, la mejor resistencia al crecimiento en complejidad y a nuevas parametrizaciones podemos encontrarla al usar AEd.

Como comentario adicional, si sólo se tomaran en consideración las primeras 20 instancias, las cuales son resolubles en casi un 100 % de los casos (ver Cuadro 5), se puede observar que el incremento en el tamaño de la población mejora la calidad para la versión en panmixia del algoritmo (AE) específicamente, y en menor medida para las otras dos variantes (AEd y AEp).

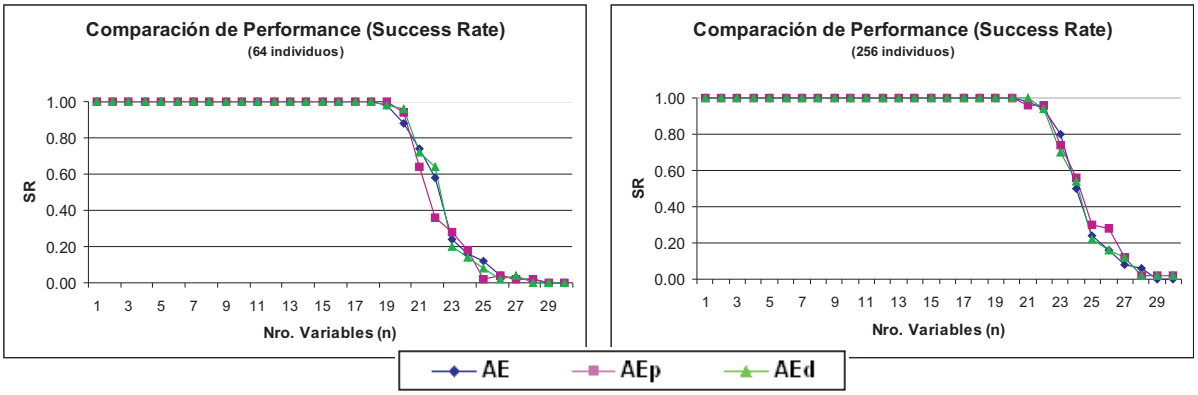

Figura 4: Análisis de SR sobre cada una de las instancias.

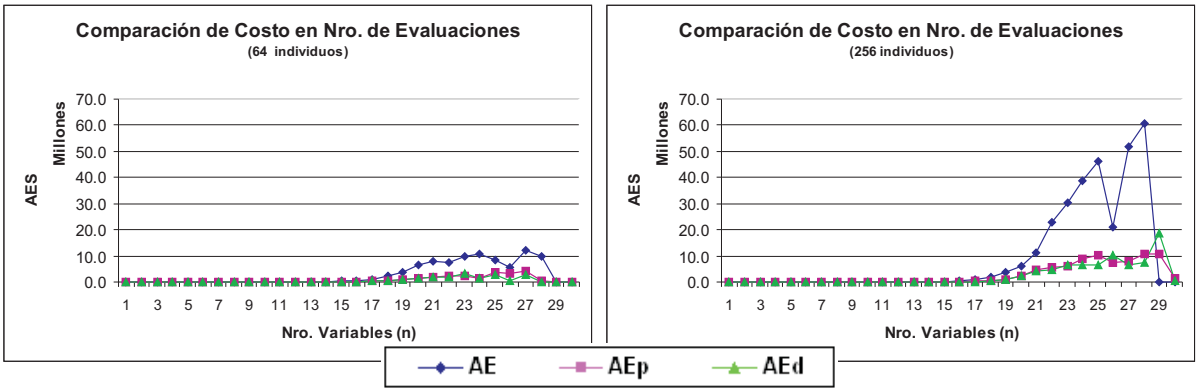

Figura 5: Análisis de AES sobre cada una de las instancias.

Lo más destacable es que el esfuerzo medido en cantidad de evaluaciones necesarias para alcanzar el óptimo (AES) no sufre una variación significativa dependiendo del tamaño de la población, e incluso podríamos decir que es cuasi constante.

Tamaño de Población	AE		AEp		AEd	
	SR	AES	SR	AES	SR	AES
64	0.93	704,706.75	1.00	173,272.68	1.00	154,846.49
128	0.96	746,023.17	1.00	188,894.61	1.00	183,135.70
256	0.99	656,427.01	1.00	205,986.37	1.00	197,569.47

Cuadro 5: Performance promedio de los algoritmos usados sobre las primeras 20 instancias considerando diversos tamaños de población.

8. Conclusiones

En este trabajo se presenta un estudio de escalabilidad de los AEs para una variante del problema 3-SAT (COUNTSAT) que cuenta con la particularidad de que la complejidad de sus instancias depende sólo de la cantidad de variables consideradas en las cláusulas y no de

cómo fueron generadas las mismas por un generador particular. COUNTSAT es un problema considerado muy duro, siendo muy difícil de alcanzar la solución óptima al tener más de 20 variables por la gran cantidad de cláusulas a satisfacer. El estudio de los AEs se ha realizado atendiendo a distintos ambientes de trabajo, para evaluar cuál de ellos presenta una mejor performance. Los ambientes considerados para ejecutar el algoritmo son el panmíctico (AE) y el distribuido en uno o cuatro procesadores (AEd o AEp, respectivamente). A partir del análisis de los resultados alcanzados se infiere que el costo del cómputo, medido en la cantidad de evaluaciones que debe realizarse para alcanzar el valor óptimo, crece en forma exponencial según se incremente la cantidad de variables. Pero las dos versiones de procesamiento distribuido (AEd y AEp) requieren un esfuerzo mucho menor que usando un algoritmo de población única (AE).

A los fines de poder evaluar si el comportamiento se mantiene frente a distintos tamaños de población se replicó la experimentación considerando una población menor (con la mitad de los individuos) y otra mayor (con el doble de individuos) de la cual podemos indicar que:

1. Los resultados de los tres experimentos se mantienen con idéntica apariencia, ello significa que las primeras 20 instancias se resuelven en un casi 99 % de los casos y las instancias del rango 20-30 decrementan este porcentaje a medida que n se incrementa.

2. Independientemente del tamaño de la población (64, 128 o 256) en el rango de instancias 1-20, el esfuerzo medido en AES es *casi* constante para cada algoritmo.

3. Según aumente el tamaño de la población SR mejora, pero sólo ligeramente.

4. Según aumente el tamaño de la población, el esfuerzo medido en AES se va *casi* duplicando.

Como trabajos futuros que surgen de este podemos mencionar evaluar si las conclusiones se mantienen frente a otros problemas derivados de SAT. También es interesante explorar la transición de dificultad de COUNTSAT y encontrar problemas con similares características para evaluar la escalabilidad de los AEs usados.

Referencias

[AAB+01] E. Alba, F. Almeida, M. Blesa, C. Cotta, M. Díaz, I. Dorta, J. Gabarró, J. González, C. León, L. Moreno, J. Petit, J. Roda, A. Rojas, and F. Xhafa. Mallba: Towards a combinatorial optimization library for geographically distributed systems. In *Proceedings of the XII Jornadas de Paralelismo*, pages 105–110. Editorial U.P.V., 2001.

[AT99] E. Alba and J.M. Troya. A survey of parallel distributed genetic algorithms. *COMPLEXITY*, 4(4):31–51, 1999.

[BEV98] Back, Eiben, and Vink. A superior evolutionary algorithm for 3-SAT. In *EP: International Conference on Evolutionary Programming, in cooperation with IEEE Neural Networks Council*. LNCS, 1998.

[Coo71] S.A. Cook. The complexity of theorem-proving procedures. *In Proc. of the 3rd. Anual ACM Symposium on the Theory of Computing*, pages 151–158, 1971.

[DJW00] Stefan Droste, Thomas Jansen, and Ingo Wegener. A natural and simple function which is hard for all evolutionary algorithms. Technical Report ISSN 1433-3325, University of Dortmund, August 2000.

[DS89] Kenneth A. De Jong and William M. Spears. Using genetic algorithm to solve NP-complete problems. In James D. Schaffer, editor, *Proc. of the Third Int. Conf. on Genetic Algorithms*, pages 124–132, San Mateo, CA, 1989. Morgan Kaufmann.

[EvdH97] A.E. Eiben and J.K. van der Hauw. Solving 3-sat with adaptive genetic algorithms. In *Proceedings of the 4th IEEE Conference on Evolutionary Computation*, pages 81–86. IEEE Press, 1997.

[FF93] Charles Fleurent and Jacques Ferland. Genetic hybrids for the quadratic assignment problem, 1993.

[FPS98] G. Folino, C. Pizzuti, and G. Spezzano. Combining cellular genetic algorithms and local search for solving satisfiability problems, 1998.

[GJ79] M. Garey and D. Johnson. *Computers and Intractability: a Guide to the Theory of NP-completeness*. Freeman, San Francisco, California, 1979.

[GMR02] Jens Gottlieb, Elena Marchiori, and Claudio Rossi. Evolutionary algorithms for the satisfiability problems. *Evolutionary Computation*, 10(2):35–50, Spring 2002.

[GPFW96] J. Gu, P. Purdom, J. Franco, and B. Wah. Algorithms for the satisfiability (sat) problem: a survey, 1996.

[GV98] J. Gottlieb and N. Voss. Improving the performance of evolutionary algorithms for the satisfiability problem by refining functions. In A.E. Eiben, T. Bck, M. Schoenauer, and H.-P. Schwefel, editors, *Parallel Problem Solving From Nature - PPSN V*, volume 1498, pages 755 – 764. Springer, 1998.

[JK] Michiel B. De Jong and Walter A. Kosters. Solving 3-sat using adaptive sampling.

[KS92] Henry A. Kautz and Bart Selman. Planning as satisfiability. In *Proceedings of the Tenth European Conference on Artificial Intelligence (ECAI'92)*, pages 359–363, 1992.

[MR99] Elena Marchiori and Claudio Rossi. A flipping genetic algorithm for hard 3-SAT problems. In Wolfgang Banzhaf, Jason Daida, Agoston E. Eiben, Max H. Garzon, Vasant Honavar, Mark Jakiela, and Robert E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 1, pages 393–400, Orlando, Florida, USA, 13-17 1999. Morgan Kaufmann.

[MS89] B. Manderick and P. Spiessens. Fine-grained parallel genetic algorithms. *Proc. 3rd International Conf. on Genetic Algorithms*, pages 428–433, 1989.

[Pap94] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[PB91] Spiessens P. and Manderick B. A massively parallel genetic algorithm. *Proceedings of the 4th International Conference on Genetic Algorithms*, pages 238–245, 1991.

[RM89] R. Reiter and A. Mackworth. A logical framework for depiction and image interpretation. *Artifitial Intelligence*, 41(3):123–155, 1989.

[RW98] Soraya Rana and Darrell Whitley. Genetic algorithm behavior in the MAXSAT domain. *Lecture Notes in Computer Science*, 1498:785–790, 1998.

[S.68] Tseitin G. S. On the complexity of derivation in propositional calculus. *Studies in Constructive Mathematics and Mathematical Logic*, 2:115–125, 1968. New York-London.

[Tan89] Reiko Tanese. Distributed genetic algorithms. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 434–439, San Mateo, CA, 1989. Morgan Kaufman.

[WS90] D. Whitley and T. Starkweather. Genitor II: A distributed genetic algorithm. *Journal of Experimental and Theoretical Artificial Intelligence*, 2:189–214, 1990.